CLAIMS:

1. A computer-implemented method for automatically updating business components information, and propagating changes in business components to other business components according to a dependency model, said method comprising:

applying rules that describe how an event affects a business component and that describe when a change in a business component triggers an event to create a business dependency model modeling business components and dependencies between them including compound dependencies, said dependency model defining event types, business component types, and dependency types associated with a business domain, said dependency model further defining how information is propagated from one business component to another; and

responsive to one or more events and/or and constraints violations, automatically updating business components information, and propagating changes in business components to other business components according to the dependency model.

2. The method according to Claim 1, further including receiving as input said event types, business component types, and dependency types associated with a business domain.

3. The method according to Claim 1, further including receiving as input rules that describe how an event affects a business component.

4. The method according to Claim 1, further including receiving as input rules that describe when a change in a business component triggers an event.

5. The method according to Claim 1, further including defining said event types, business component types, and dependency types associated with a business domain.

6. The method according to Claim 1, further including defining said rules that describe how an event affects a business component.

7. The method according to Claim 1, further including defining said rules that describe when a change in a business component triggers an event.

8. The method according to Claim 1, wherein the business dependency model includes predefined dependency type semantics.

9. The method according to Claim 8, wherein said dependency type semantics include a "mandatory" logical operator that logically couples one or more source components of the dependency to one ore more targets of the dependency and sets the targets to a worst state of the sources.

10. The method according to Claim 8, wherein said dependency type semantics include an "N out of M" logical operator that logically couples M source components of the dependency to one or more targets of the dependency and sets the targets to ok if at least N of the sources are ok and otherwise sets the targets to "fail".

11. A program storage device readable by machine, tangibly embodying a program of instructions executable by the machine to perform method steps for automatically updating business components information, and propagating changes in business components to other business components according to a dependency model, said method comprising:

applying rules that describe how an event affects a business component and that describe when a change in a business component triggers an event to create a business dependency model modeling business components and dependencies between them including compound dependencies, said dependency model defining event types, business component types, and dependency types associated with a business domain, said dependency model further defining how information is propagated from one business component to another; and

responsive to one or more events and/or and constraints violations, automatically updating business components information, and propagating changes in business components to other business components according to the dependency model.

12. A computer program product comprising a computer useable medium having computer readable program code embodied therein for automatically updating business components information, and propagating changes in business components to other business components according to a dependency model, said computer program product comprising:

computer readable program code for causing the computer to apply rules that describe how an event affects a business component and that describe when a change in a business component triggers an event to create a business dependency model modeling business components and dependencies between them including compound dependencies, said dependency model defining event types, business component types, and dependency types associated with a business domain, said dependency model further defining how information is propagated from one business component to another; and

computer readable program code for causing the computer to automatically update business components information, and to propagate changes in business components to other business components according to the dependency model in response to one or more events and/or and constraints violations.

13. A system for automatically updating business components information, and propagating changes in business components to other business components according to a dependency model, said system comprising:

a model generator that applies rules that describe how an event affects a business component and that describe when a change in a business component triggers an event to create a business dependency model modeling business components and dependencies between them including compound dependencies, said dependency model defining event types, business component types, and dependency types associated with a business domain, said dependency model further defining how information is propagated from one business component to another; and

an active dependency integration unit responsive to one or more events and/or and constraints violations for automatically updating business components information, and propagating changes in business components to other business components according to the dependency model.

14. The system according to Claim 13, wherein the model generator includes an input for receiving said event types, business component types, and dependency types associated with a business domain.

15. The system according to Claim 13, wherein the model generator includes an input for receiving said rules that describe how an event affects a business component.

16. The system according to Claim 13, wherein the model generator includes an input for receiving said rules that describe when a change in a business component triggers an event.

17. The system according to Claim 13, wherein the model generator is coupled to a user interface for allowing manual input of said event types, business component types, and dependency types associated with a business domain.

18. The system according to Claim 13, wherein the model generator is coupled to a user interface for allowing manual input of said rules that describe how an event affects a business component.

19. The system according to Claim 13, wherein the model generator is coupled to a user interface for allowing manual input of said rules that describe when a change in a business component triggers an event.

20. The system according to Claim 13, wherein the model generator is coupled to a user interface for allowing manual input of said rules that describe when a change in a business component triggers an event.

21. The system according to Claim 13, wherein the active dependency integration unit is responsively coupled to a situation awareness unit for receiving therefrom new situations resulting from events triggered by a change in a business component.

22. The system according to Claim 13, wherein the model is an XML format.

23. The system according to Claim 13, wherein the model is loaded on startup under control of suitable APIs.

24. The system according to Claim 13, wherein the business dependency model includes predefined dependency type semantic.

25. The system according to Claim 24, wherein said dependency type semantics include a "mandatory" logical operator that logically couples one or more source components of the dependency to one ore more targets of the dependency and sets the targets to ok if all the sources are ok and sets the targets to a worst state of the sources.

26. The system according to Claim 24, wherein said dependency type semantics include an "N out of M" logical operator that logically couples M source components of the dependency to one or more targets of the dependency and sets the targets to ok if at least N of the sources are ok and otherwise sets the targets to "fail".